Inherently Interpretable Models[1]

October 2, 2023

Linwei Hu

Advanced Technologies for Modeling Group
Corporate Model Risk

[1] This material represents the views of the presenter(s) and does not necessarily reflect those of Wells Fargo

Outline

- Examples of Inherently Interpretable Models (IIM)
- Functional ANOVA models: 2nd order
  - Explainable boosting machine (EBM)
  - GAMI-Net
  - GAMI-Lin-Tree
  - Monotone GAMI-Tree
- Other IIM: MARS, FIGS, rulefit
- Illustrative example
- Summary

Inherently interpretable models

- Key characteristics
  - **Parsimony** → easier to interpret
    - ✓ **Sparsity** → few active effects or complicated relationships
    - ✓ **Low-order interactions** → more than two hard to understand
  - **Analytic expression** → use **regression coefficients** for interpretation
- Goals and challenges of complex ML models
  - **Extract as much predictive performance** as possible
  - **No emphasis on interpretation** → lots of variables, complex relationships and interactions
  - No analytic expressions → **rely on low dimensional summaries** → **don't present the full picture**
- Emerging view
  - **Low-order functional** (nonparametric) **models** are **adequate** in most of our applications (Lou et al. 2013)
  - **Directly interpretable**
  - **Reversing emphasis on complex modeling**

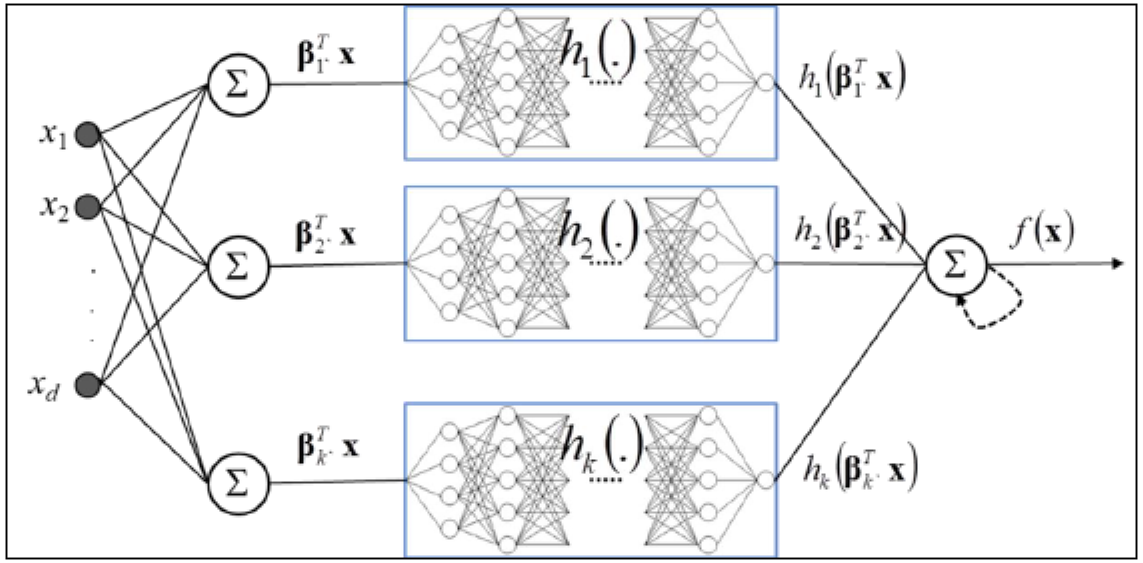
    → **trade-off**: small improvements in **predictive performance vs interpretation**

Example of IIM

- **Parametric models**
- **Small decision trees.** Can be represented by a set of rules
- **Additive Index Models:**

$$f(\boldsymbol{x}) = g_1(\boldsymbol{\beta}_1^T\boldsymbol{x}) + g_2(\boldsymbol{\beta}_2^T\boldsymbol{x}) + \cdots + g_K(\boldsymbol{\beta}_k^T\boldsymbol{x})$$

  - Generalization of Generalized Additive Models (GAMs): $f(\boldsymbol{x}) = g_1(x_1) + g_2(x_2) + \cdots + g_K(x_p)$
  - Incorporates certain types of interactions
  - **Projection pursuit regression** (Friedman and Stuetzle, 1981)
  - **Need for scalable algorithms** with large datasets and many predictors
  - Use specialized **neural network architecture and associated fast algorithms**
    - e**X**plainable**N**eural **N**etworks (xNNs) →Vaughan, Sudjianto, … Nair (2020)


x_1
x_2
x_d
Σ
$\boldsymbol{\beta}_1^T \mathbf{x}$
$\boldsymbol{\beta}_2^T \mathbf{x}$
$\boldsymbol{\beta}_k^T \mathbf{x}$
$h_1(.)$
$h_2(.)$
$h_k(.)$
$h_1(\boldsymbol{\beta}_1^T \mathbf{x})$
$h_2(\boldsymbol{\beta}_2^T \mathbf{x})$
$h_k(\boldsymbol{\beta}_k^T \mathbf{x})$
$f(\mathbf{x})$

Example of IIM

- **Functional ANOVA Models:**

$$f(\boldsymbol{x}) = g_0 + \sum_{j} g_j(x_j) + \sum_{j<k} g_{jk}(x_j, x_k) + \sum_{j<k<l} g_{jkl}(x_j, x_k, x_l) + \cdots$$

  - FANOVA models with low-order interactions are adequate for many of our applications
  - **Focus** on models with **functional main effects and second order interactions**
  - Stone (1994); Wahba and her students (see Gu, 2013)

    → use **splines** to estimate low-order functional effects non-parametrically
  - **Not scalable** to large numbers of observations and predictors
  - Recent approaches

    → use **ML architecture and optimization algorithms** to develop fast algorithms
  - Focus of today's presentation

Second order FANOVA models

$$f(\boldsymbol{x}) = g_0 + \sum_j g_j(x_j) + \sum_{j<k} g_{jk}(x_j, x_k)$$

- Second order FANOVA models are called "GAMI" (GAM+interaction) Model. It is made up of mean g_0, **main effects $g_j(x_j)$, two-factor interactions $g_{jk}(x_j, x_k)$**
- **Interpretability**
  - Fitted model is **additive,** effects are enforced to be **hierarchical orthogonal**
  - Components can be **easily visualized** and **interpreted directly.** In case of tree-based models, $g_j(x_j)$ can be represented as a vector and $g_j(x_j, x_k)$ a 2-d table.
  - Regularization or other techniques used to keep model parsimonious

- Three state-of-the-art ML algorithms for fitting these models:
  - **Explainable Boosting Machine** (Nori, et al. 2019) → boosted tress
  - **GAMI Neural Networks** (Yang, Zhang and Sudjianto, 2021) → specialized NNs
  - **GAMI-Lin-Tree** (Hu, Chen, and Nair, 2022) → specialized boosted model-based trees

Nori, Jenkins, Koch and Caruana (2019). InterpretML: A Unified Framework for Machine Learning Interpretability. arXiv: 1909.09223
Yang, Zhang and Sudjianto (2021, Pattern Recognition): GAMI-Net. arXiv: 2003.07132

GAMI-models

- Different GAMI algorithms all have the following key ingredients:
  - An interaction filtering step to pre-select top k interactions, to reduce the amount of computation and have a simpler model
  - A “base model” to fit main-effects and two-way interactions non-parametrically.

- The kind of interaction filtering algorithm, or the “base model” are what differentiate the GAMI models.

Explainable boosting machine (EBM)

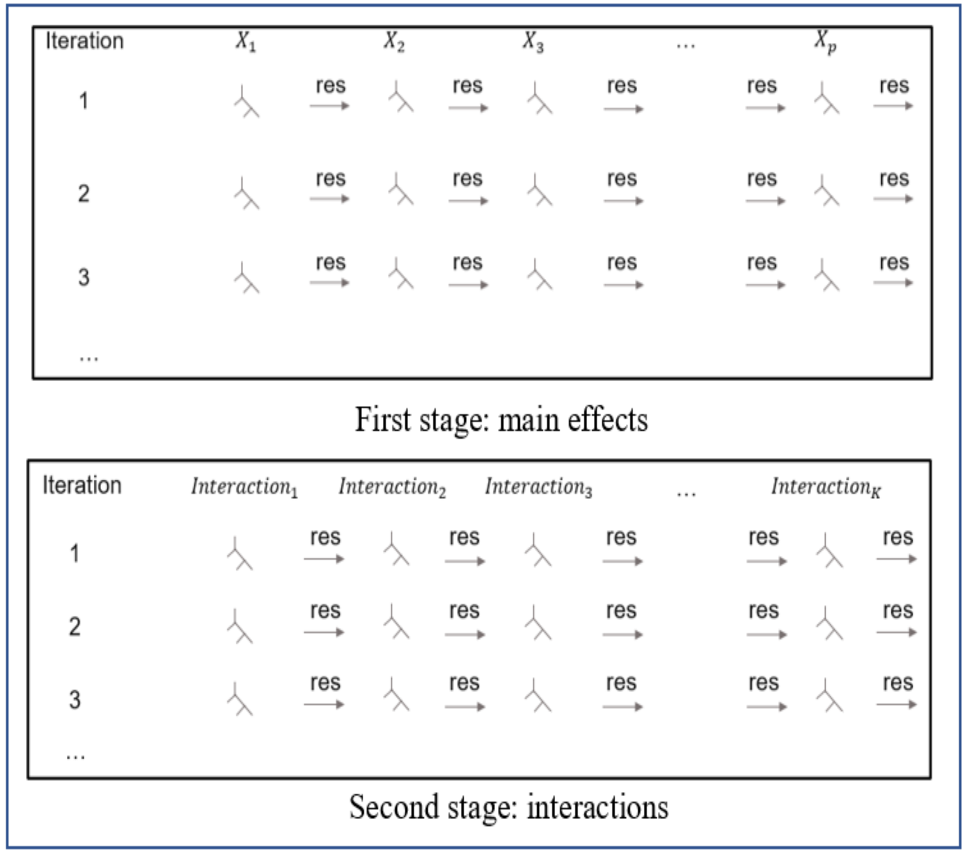
- **EBM** – Boosted-tree algorithm by Microsoft group (Lou, et al. 2013)

$$f(\boldsymbol{x}) = g_0 + \sum g_j(x_j) + \sum g_{jk}(x_j, x_k)$$

  – Microsoft InterpretML (Nori, et al. 2019)
  – fast implementation in C++ and Python

- **Two-stage model training :**

  1. fit functional main effects non-parametrically
     – Shallow tree boosting with splits on the same variable for capturing a non-linear main effect
     – loop through all main-effects until convergence
  2. fit pairwise interactions on residuals
     a. Detect top K interactions using **FAST** algorithm (→ next page)
     b. For each interaction (x_j, x_k), fit function $g_{jk}(x_j, x_k)$ non-parametrically using a tree which cuts only on x_j and x_k
     c. Loop through all the detected interactions until convergence



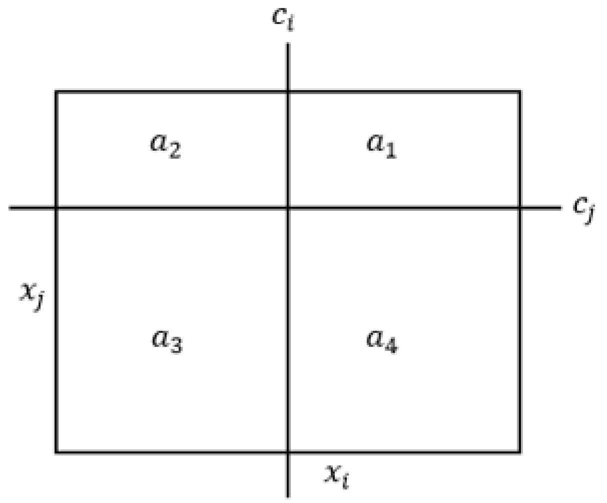
First stage: main effects

Second stage: interactions

Lou, Caruana, Gehrke and Hooker (2013). Accurate Intelligible Models with Pairwise Interactions. Microsoft Research

EBM: FAST algorithm

- To select top K interactions, EBM uses an algorithm called FAST.
- **FAST:**
  - Compute the residuals after fitting the main-effect stage. This makes sure only interactions are left.
  - For each pair of variables (x_i, x_j):
    - Select one cut point (from a set of m candidate cut points) for each variable, divide the 2-d space into 4 quadrants
    - Fit a constant in each quadrant, a_1, a_2, a_3, a_4 (4-quadrant model)
    - Compute the sum of square error
    - Find the best cut points (c_i, c_j) which minimizes SSE
  - Rank all pairs according to SSE and select top K. Total computation: $\binom{p}{2} \times m^2$ 4-quadrant models to fit and evaluate!
- FAST uses the simple 4-quadrant model to approximate interaction. Lou et al. (2013) justified this approximation by arguing that fully building the interaction structure for each pair “is a very expensive operation”.



Interpretation

- Global importance: each **effect importance** (before normalization) is given by

$$D(h_j) = \frac{1}{n-1}\sum_{i=1}^{n} g_j^2(x_{ij}), \qquad D(f_{jk}) = \frac{1}{n-1}\sum_{i=1}^{n} g_{jk}^2(x_{ij}, x_{ik})$$

- Local explanation: for prediction at $\boldsymbol{x}_i$, the **local feature importance** is given by (in piML)

$$\phi_j(x_{ij}) = g_j(x_{ij}) + \frac{1}{2}\sum_{j\neq k} g_{jk}(x_{ij}, x_{ik})$$

  - The coefficient ½ means to distribute the interaction term $g_{jk}(x_{ij}, x_{ik})$ equally to each variable.
  - Seems reasonable, but what about SHAP? It can be shown under variable independence, this is exactly kernel SHAP (Hu and Wang, 2023)
  - SHAP can be used instead. SHAP computation is very efficient for such GAMI-models (Hu and Wang, 2023).
- The effects can be visualized by a line plot (for main effect) or heatmap (for pairwise interaction).

Hu and Wang (2023). Computing SHAP efficiently using model structure information. arxiv-2309.02417.

EBM: example output

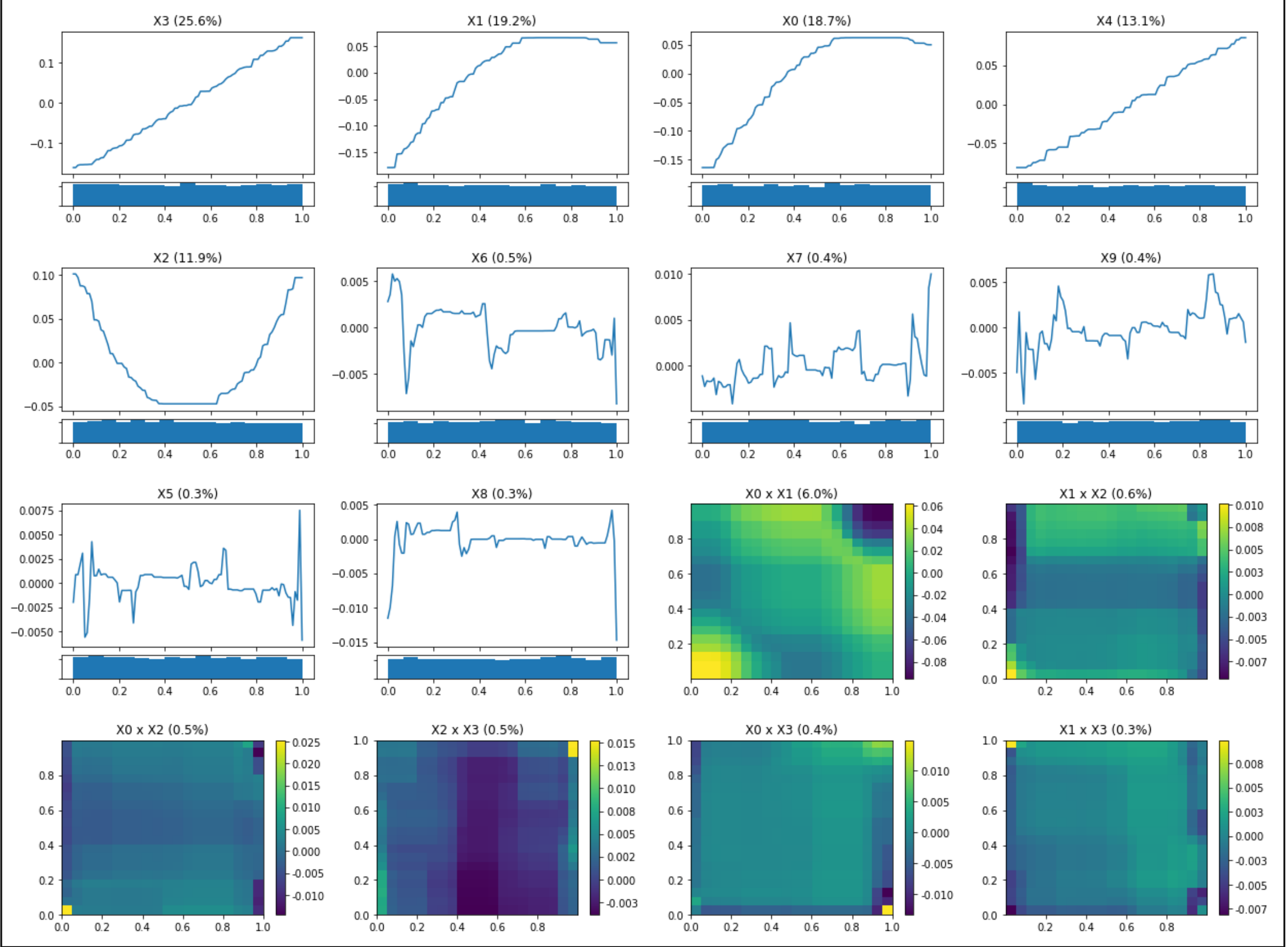
Friedman1 simulated data:

- sklearn.datasets.make_friedman1 n_samples=10000, n_features=10, and noise=0.1.
- Multivariate independent features $\boldsymbol{x}$ uniformly distributed on [0,1]
- Continuous response generated by

$$y(\boldsymbol{x}) = 10\sin(\pi x_0 x_1) + 20(x_2 - 0.5)^2 + 20x_3 + 10x_4 + \epsilon$$

depending only $\boldsymbol{x_0}\sim\boldsymbol{x_4}$

EBM Output with Test RMSE = 0.0284 and R2 = 97.39%

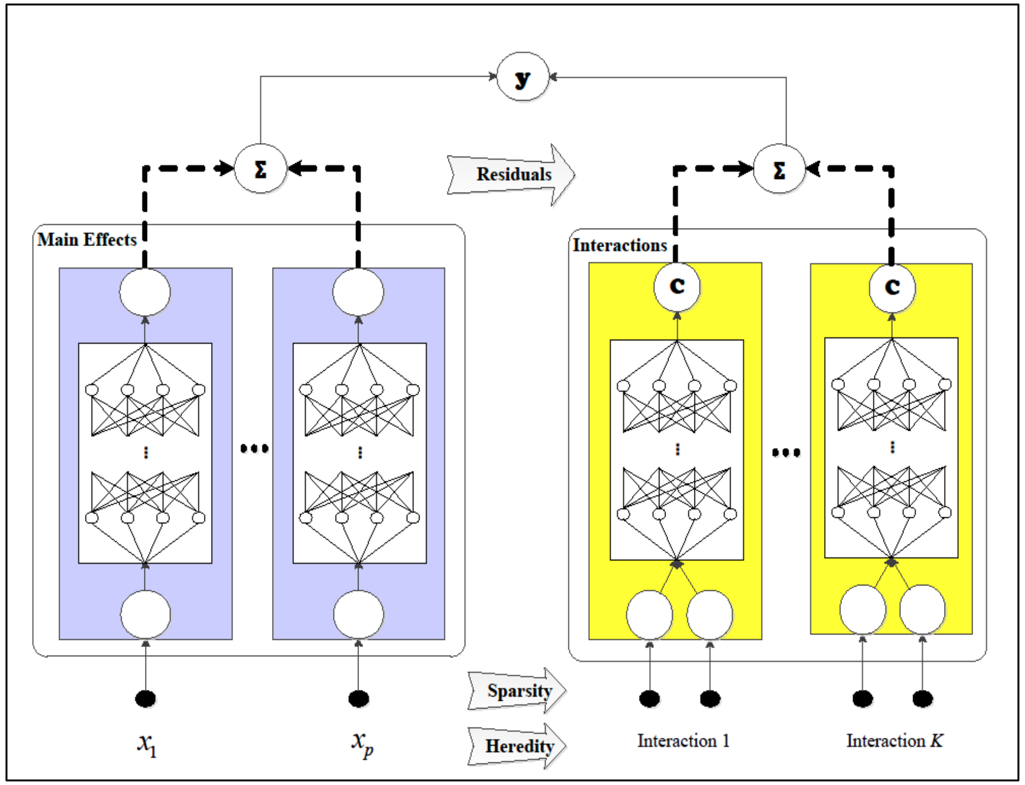


GAMI-Net

- NN-based algorithm for non-parametrically fitting

$$f(\boldsymbol{x}) = g_0 + \sum g_j(x_j) + \sum g_{jk}(x_j, x_k)$$

- Interaction effect filtering: using FAST
- Main-effect modeling: GAM subnet. GAM-subset is a fully connected sub-network that only use one single variable as input. It only models main-effect of that variable.
- Interaction-effect modeling: Interaction subset. Similar to GAM subset but uses two variables as input.
- Small main-effects or interaction-effects are pruned.
- Eventually, important GAM subnet and interaction subnets are combined and trained jointly.



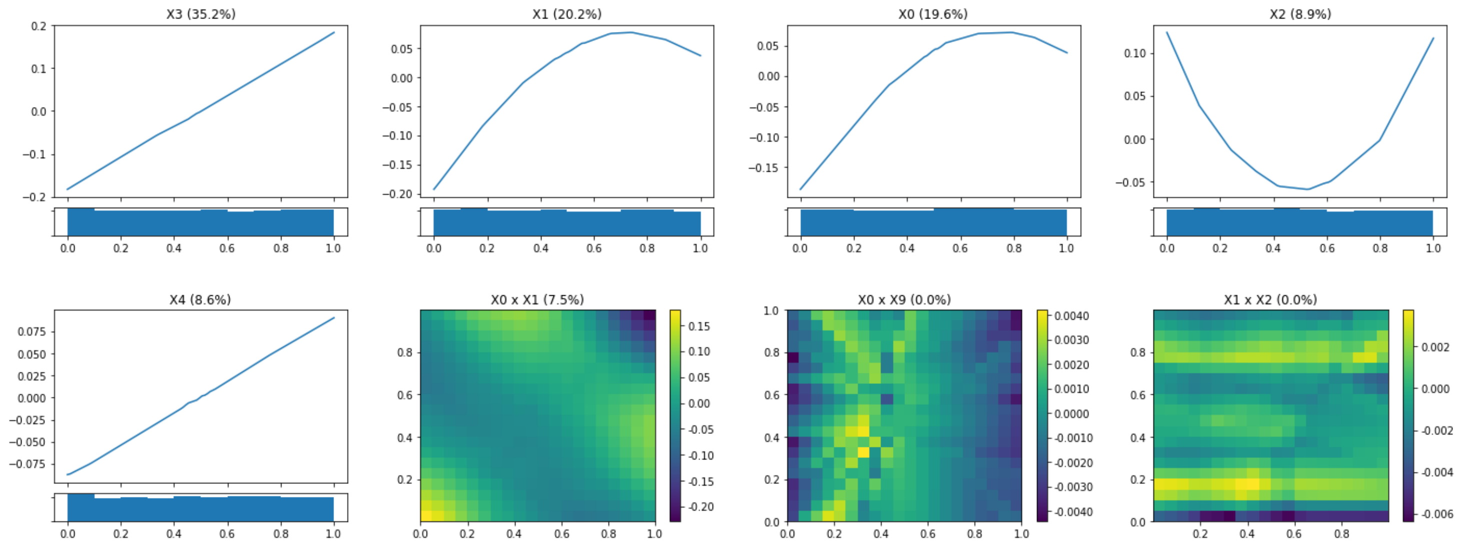
Yang, Zhang and Sudjianto (2021, Pattern Recognition): GAMI-Net. arXiv: 2003.07132

GAMI-Net: Example

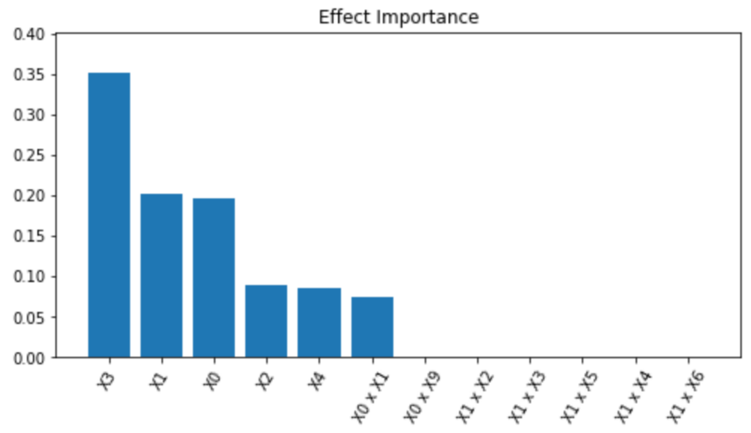
Friedman1 data:

$$y(\boldsymbol{x}) = 10\sin(\pi x_0 x_1) + 20(x_2 - 0.5)^2 + 20x_3 + 10x_4 + \epsilon$$

GAMI-Net Output with Test RMSE = 0.0058 and R2 = 99.89%

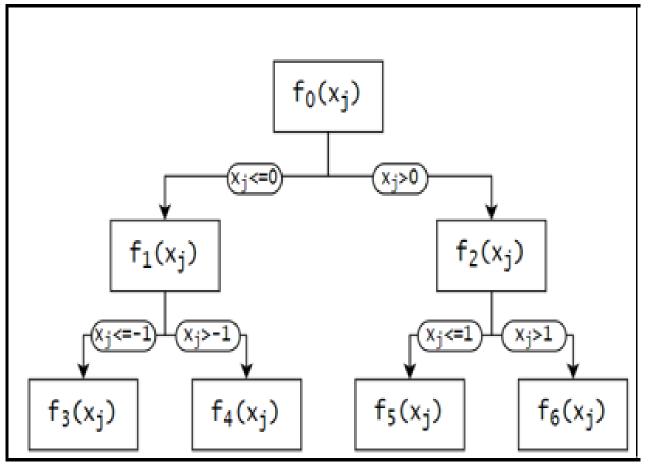


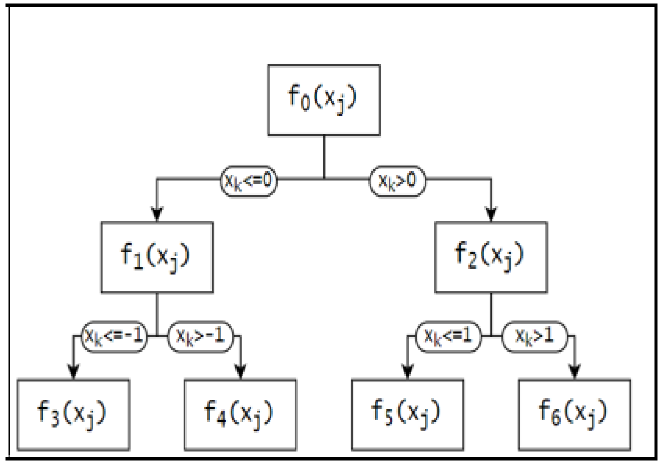
```
model_gaminet.show_effect_importance()
```



GAMI-Lin-Tree

- GAMI-Lin-Tree (Hu, Chen, Nair 2022) is similar to EBM, with the following differences:
  - it uses linear/spline model-based tree instead of piece-wise constant trees as base learner.
  - It uses a more accurate interaction filtering method
  - It iterates between main-effect and interaction stages.
- Model-based tree fits a model in each tree node, instead of a constant.
- Main-effect tree $T(x_j)$:
  - Use same variable x_j for both tree split and modeling inside the node.
  - Model $f_v(x_j)$ at node v is chosen as a simple linear model
  - Splits account for nonlinearity
- Interaction tree $T(x_j, x_k)$:
  - Use one variable to split the tree and another variable to model inside each node.
  - Model $f_v(x_j)$ in node v is chosen as linear B-spline model to allow more flexibility.
  - Splits account for interaction





Hu, Linwei, Jie Chen, and Vijayan N. Nair. 2022. "Using Model-Based Trees with Boosting to Fit Low-Order Functional ANOVA Models." arXiv:2207.06950

GAMI-Lin-Tree

- The interaction tree is also used for interaction filtering. It is a more flexible, more accurate model than the 4-quandrant model used in EBM.
- GAMI-Lin-Tree iterates between fitting main-effects and interactions until the model converges, instead of fitting only one main-effect stage and one interaction stage. So the final model can converge to a better model than EBM.

Comparison: EBM vs GAMI-Lin-Tree vs GAMI-Net

- Hu, Chen, Nair (2022) compared EBM, GAMI-Lin-Tree and GAMI-Net.
- GAMI-Lin-Tree and GAMI-Net have close model performance, both are comparable or better than EBM in some cases.
  - The two-stage algorithm used in EBM may not converge to the best model. In such case, the fine-tuning stage in GAMI-Net and the iterative training in GAMI-Lin-Tree converges to better model.
- The FAST interaction filtering method works well in general, but it could miss weaker interactions, or interactions which have highly nonlinear pattern. In such case, the interaction filtering method used in GAMI-Lin-Tree is more accurate.
- When FAST misses interaction, it affects the model performance of GAMI-Net and EBM.
- In terms of overfitting, GAMI-Net has least train-test gap because it is smooth. GAMI-Lin-Tree and EBM are not smooth.

Monotonicity

- Monotonicity is a very important requirement. A variable x_j is monotone increasing if $f\left(x_j, \boldsymbol{x}_{-j}\right)$ is monotone increasing in x_j, when holding $\boldsymbol{x}_{-j}$ fixed.
- For example, customer with longer credit history should have a higher chance of obtaining a loan, holding other factors fixed.
- To impose monotonicity on GAMI-Net, tensorflow lattice can be used.
- However, it is hard to impose monotonicity on EBM or GAMI-Lin-Tree, due to the sequential fitting of main-effect and interaction-effect.
  - Do we impose monotonicity only on main-effects? Or both main and interaction? The former is not enough, and the latter is too restrictive.
- Monotone GAMI-Tree is a GAMI-style model proposed in Hu et al. (2023), it uses two useful features in xgboost: interaction constraint and monotonicity constraint.

Monotone GAMI-Tree

- The interaction constraint in xgboost allows user to only model certain interaction effects. So one can use an interaction filtering algorithm to find the important interactions, then pass these in as a constraint.
- Xgboost also comes with monotonicity constraint. So a monotone gami-model can be fitted easily.
- However, xgboost has hundreds of trees, how to make it interpretable?
- To make the model interpretable, we do parsing and purification to obtain the main-effects and interactions FANOVA decomposition. Then we can measure and visualize the effects.
  - Parsing:
    - the xgboost model can be written as sum over all leaf node rules, $\lambda \sum_i \sum_j R_{ij}(x)$, where $R_{ij}(x)$ is the rule for tree i, leaf node j.
    - Due to interaction constraint, each rule is either a main-effect, or a 2-way interaction. So the model can be rewritten as $\sum_j g_j(x_j) + \sum_{i,j} g_{ij}(x_i, x_j)$.
  - Purification: the effects $g_j(x_j)$ and $g_{ij}(x_i, x_j)$ are not uniquely defined. Per Hooker (2007), a purification step can be added so they are uniquely defined and orthogonal.
- The effects $g_j(x_j)$ and $g_{ij}(x_i, x_j)$ can be represented as 1-d or 2-d tables, so the model can have an explicit expression.

Monotone GAMI-Tree

- To summarize, monotone GAMI-Tree is a tree-based GAMI-model, just like EBM.
- However, GAMI-Tree does not use the two-stage (main+interaction) sequential training method used in EBM. It directly uses xgboost (with interaction constraints) to train the model.
- We use parsing and purification on GAMI-Tree to achieve the same level of interpretability as EBM.
- The model performances of EBM and GAMI-Tree are close. In some cases, Hu et al. (2022) paper found that the two-stage sequential training method in EBM does not lead to a final converged model, so GAMI-Tree is better in such cases.
- Most importantly, GAMI-Tree can easily allow for monotonicity constraint, a key modeling requirement.
- We recommend monotone GAMI-Tree over EBM for modeling.

Higher order FANOVA?

- Higher order FANOVA model is possible, but much harder:
  - Interaction filtering. Filtering k-way interaction requires $\binom{p}{k} \times m^k$ combinations of variables and cut-points evaluations. The computation grows exponentially.
  - However, if the top interactions are selected, then it is quite convenient to extend the current GAMI-Net and GAMI-Tree algorithm to fit a higher order FANOVA model.
  - Interpretation. No easy way to visualize a 3-way or higher order interaction, and it is hard to interpret.

Other interpretable method: MARS

- MARS (multivariate adaptive regression spline) is a method proposed by Friedman in 1991.
- It is a non-parametric spline method that can fit model up to a certain number of terms, or a certain order of interaction. The final model has an explicit form solution: $f(x) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X)$, where $h_m(X)$ is a hinge function $\left(x_j - t\right)_+$, or a product of multiple hinge functions.
- If the order is limited at 2, then it can fit another style of GAMI models, using hinge splines as basis functions.
- The algorithm starts with only the intercept term $h_0(X) = 1$, then
  - Forward selection: adding the product of a new basis function, $\left(x_j - t\right)_+$, with all existing model terms. The basis function which can reduce SSE most is selected. Keep adding until a specified max number of terms is reached.
  - Backward elimination: drop terms which are least important.

Other interpretable method: FIGS

- Fast Interpretable Greedy-tree Sums (FIGS; [Tan2022]) is a recently proposed machine-learning algorithm that extends classification and regression trees (CART).
- FIGS can also be viewed as a special case boosted tree model, as a fitted FIGS usually consists of multiple additive trees.
- However, unlike XGBoost, FIGS can decide whether to boost a new tree or grow existing trees, whichever reduces the loss most.
- As a result, “[FIGS] overcome a key weakness of single-tree models by disentangling additive components of generative additive models, thereby reducing redundancy from repeated splits on the same feature”.

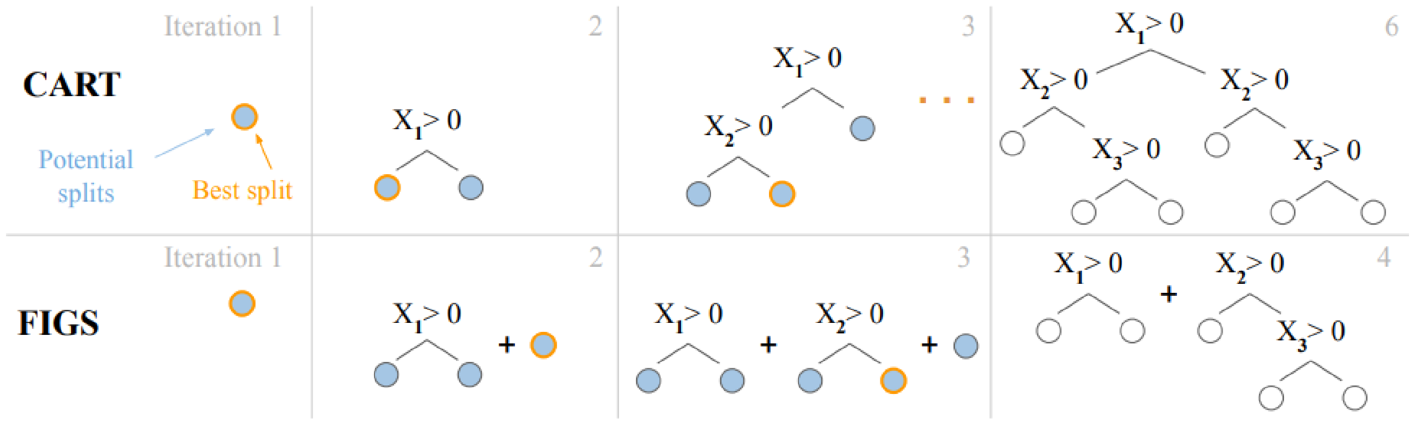


Figure 1. FIGS algorithm overview for learning the toy function $y = \mathbb{1}_{X_1>0} + \mathbb{1}_{X_2>0} \cdot \mathbb{1}_{X_3>0}$. FIGS greedily adds one node at a time, considering splits not just in an individual tree but within an ensemble of trees. This can lead to much more compact models, as it avoids repeated splits (e.g. in the final CART model shown in the top-right).

Other interpretable method: rulefit

- Rulefit (Friedman and Popescu 2008) learns a sparse linear model using rules learned from decision tree.
- Rules are generated using paths in decision tree
- A sparse linear model is fitted using the rules. L1 penalty is used to select rules.
- Rulefit automatically adds feature interactions to linear models, and the rules are easy to interpret. However, the interpretability decreases with the number of rules selected.

GAMI-model Illustrative Example

Bike Sharing Data

Bike sharing data:

- A popular benchmark UCI dataset consisting of hourly count of rental bikes between years 2011 and 2012 in Capital bikeshare system in DC.
- Sample size: 17379 observations
- The features include weather conditions, precipitation, day of week, season, hour of the day, etc.
- The response is *log* count of total rental bikes.

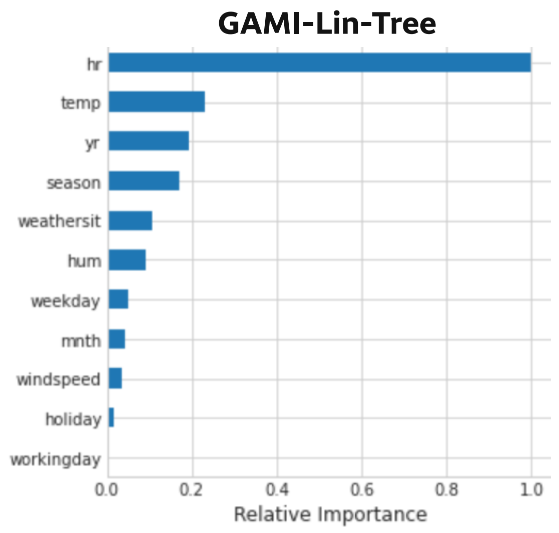
Model fitting

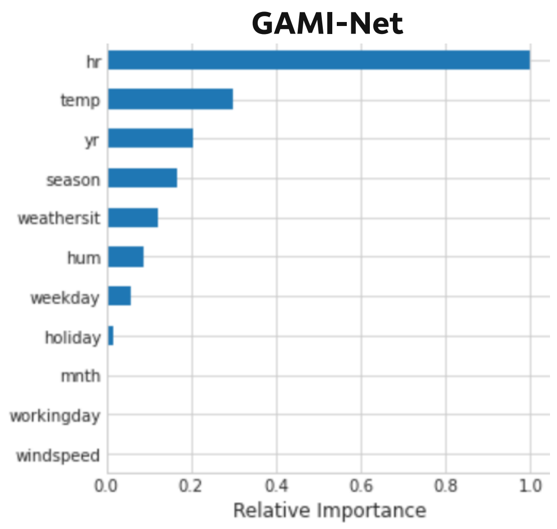
- The data is split into 50% train, 25% validation, and 25% testing. The following algorithms were fit: (unconstrained) xgboost, GAMI-Net, GAMI-Lin-Tree and EBM.
- The models are tuned using the validation set. The model performance is assessed on testing set.
- We can see the GAMI models perform close to xgboost, while xgboost is slightly better.

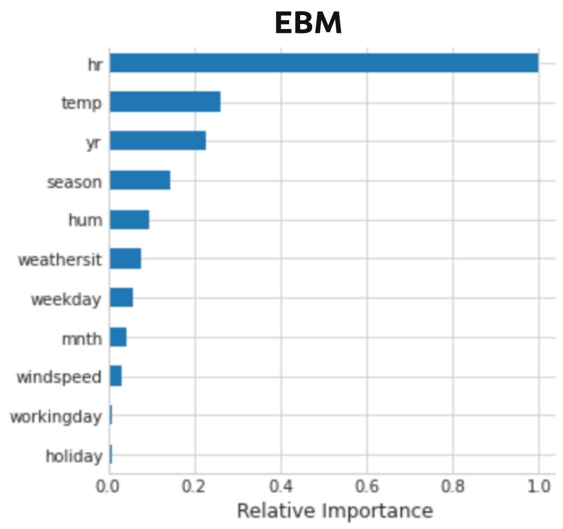
	xgboost	GAMI-Net	GAMI-Lin-Tree	EBM
test_R2	0.955	0.946	0.953	0.952
test_mse	0.099	0.119	0.103	0.107

Model interpretation

- The importance of main-effects from GAMI-Lin-Tree, GAMI-Net and EBM are shown below.
- All algorithms yield similar rankings with some slight change of orders.
- Recall in the demo last class, hr is the most important variable in terms of PFI, followed by workingday. Here, working day is ranked low because it only accounts for main-effects, whereas PFI accounts for both main-effect and interactions.

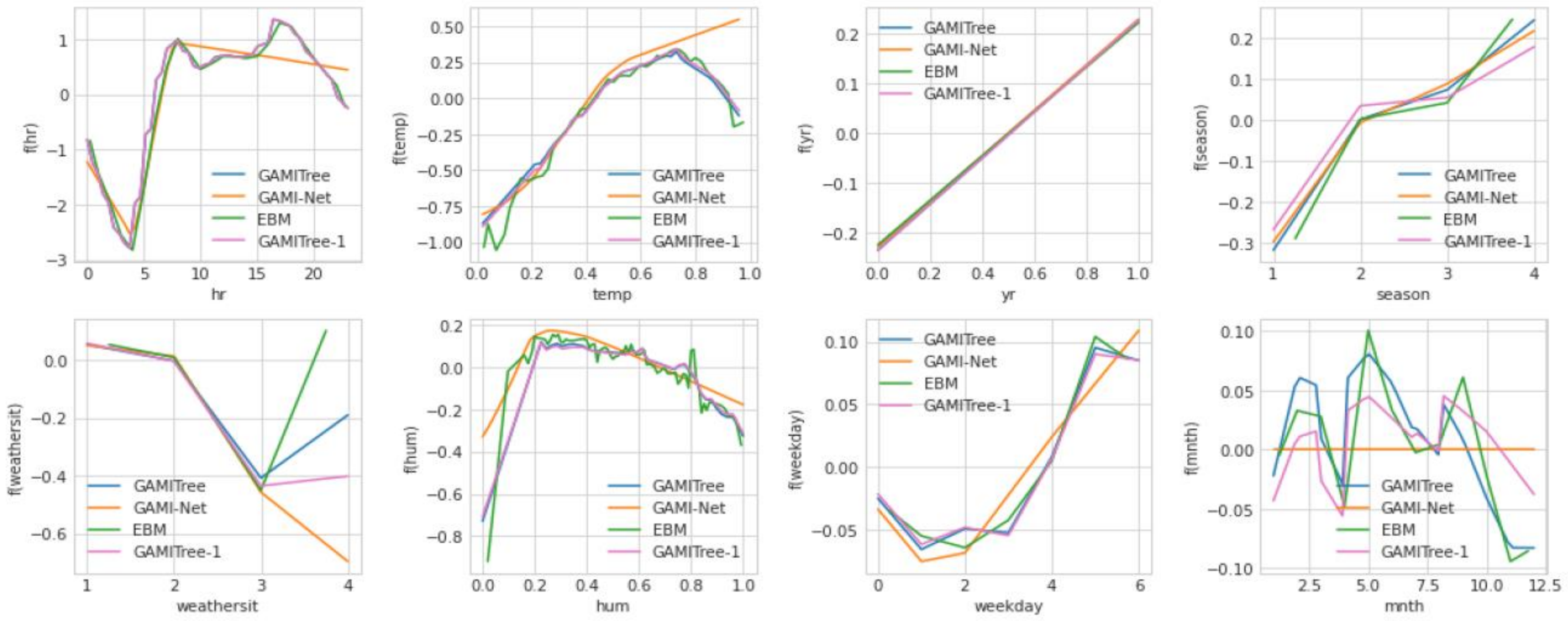






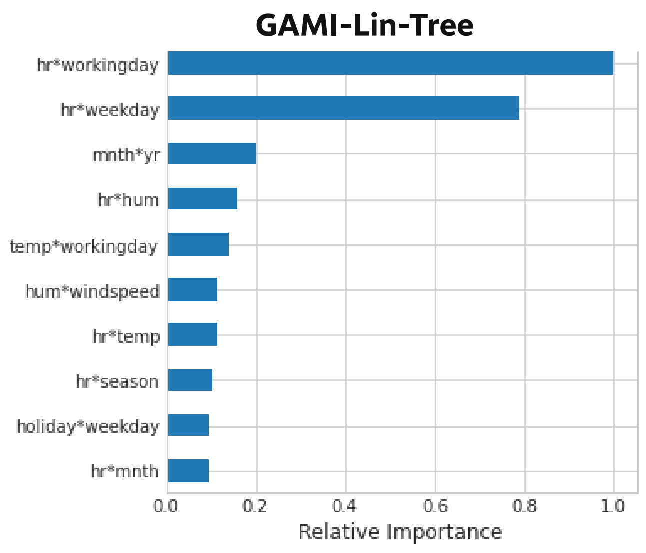
Model interpretation

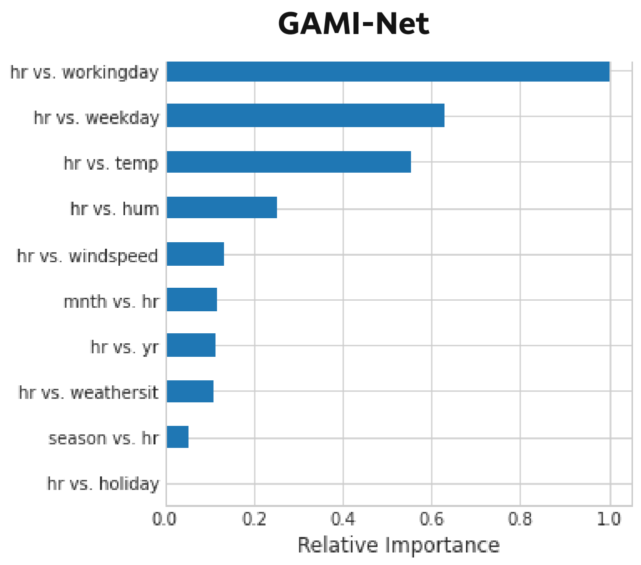
- The main-effects can be plotted and visualized (legend GAMITree -> GAMI-Lin-Tree)
- We see all 3 algorithms have similar main-effects. Some differences are:
  - GAMI-Net only shows one peak, instead of one in the am and one in the pm. We will explain it later
  - When Weathersit = 4, we see big differences. However, weathersit=4 is very rare.

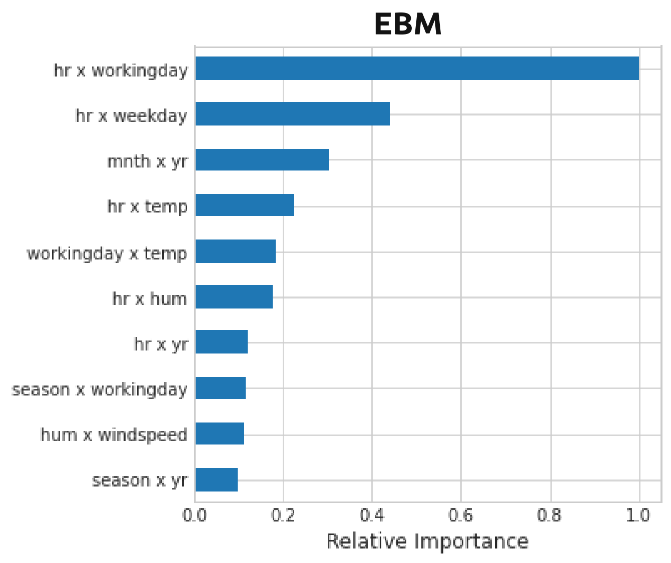


Model interpretation

- The top 10 interactions are given below.
- The top two pairs identified by all three are the same. The other interactions are smaller and not consistent among all algorithms. Eg, GAMI-Lin-Tree and EBM has the same 3rd most important interactions, GAMI-Lin-Tree and GAMI-Net has the same 4th most important interactions.

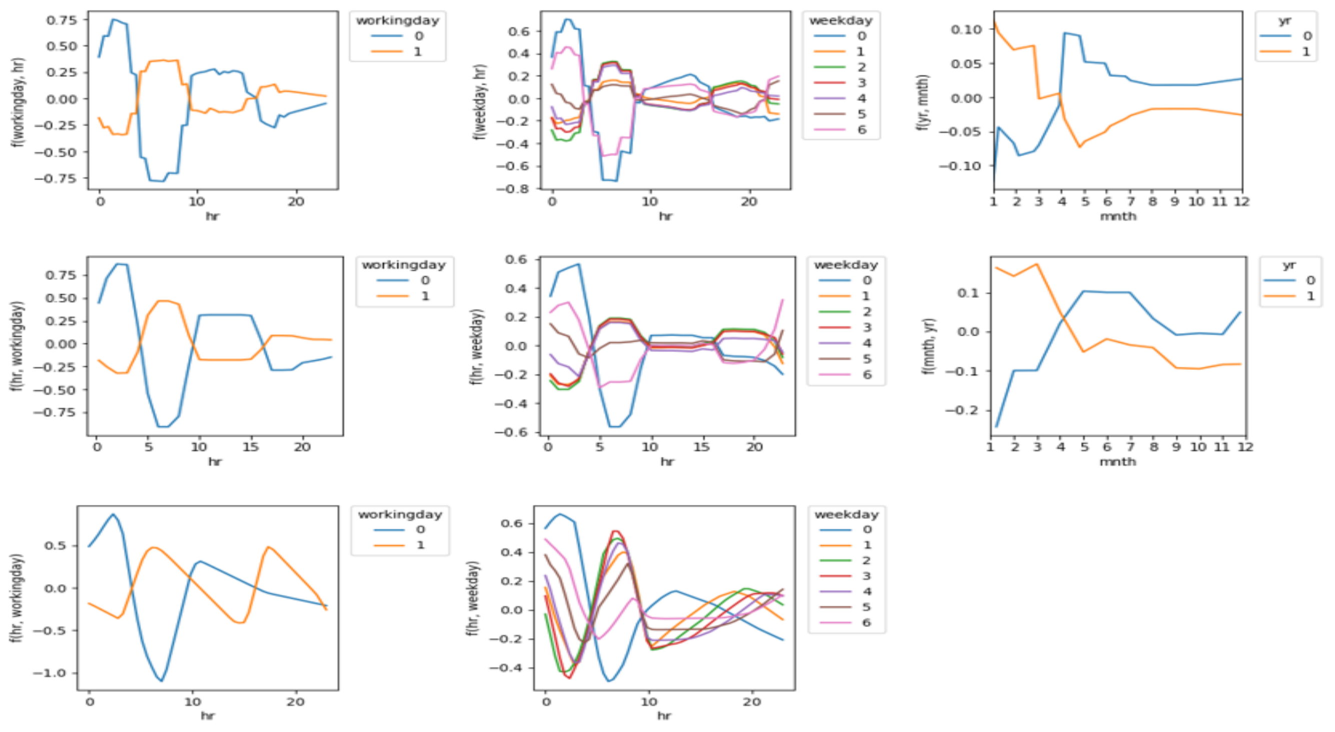






Model interpretation

- The top 3 interactions for GAMI-Lin-Tree (top), EBM (middle) and GAMI-Net (bottom)
- We can see for hr and working day interaction:
  - There are relatively more bike rentals in mid-night and 10-15 when working day is 0 compared to 1
  - For GAMI-Net, the double peak in hour for workday is more pronounced than GAMI-Lin-Tree or EBM. This explains why the main-effect of hour only has 1 peak for GAMI-Net. i.e., slightly different decomposition of main-eff and interaction!



Summary

- We introduced different IIM models.
- One type of IIM model that has seen increasing popularity is the 2nd order FANOVA models (GAMI-models).
- The GAMI-models have good predictive performance and can be interpreted directly.
- Different ML algorithms have been implemented to fit such models, which is fast and scalable.
- Monotonicity can be imposed on some GAMI-models, like GAMI-Net and GAMI-Tree.